PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

EX PARTE Shridhar Mukund et al.

Application for Patent

Filed November 12, 2003

Application No. 10/712,711

FOR:

SIMULATION OF COMPLEX SYSTEM ARCHITECTURE

APPEAL BRIEF

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on July 20, 2009.

Signed: Justine Stamm
Justine Stamm

MARTINE PENILLA & GENCARELLA, LLP
Attorneys for Applicants

TABLE OF CONTENTS

Page No.

I. REAL PARTY IN INTEREST

The real party in interest is Adaptec, Inc., the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

The Applicants are not aware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-19 are pending in the subject application. Claims 1-19 have been rejected and are on appeal.

IV. STATUS OF AMENDMENTS

Applicants submitted a Request for Reconsideration on February 27, 2007, in response to a Final Office Action (hereinafter, Final Office Action) mailed on November 29, 2006. Per the Advisory Action dated March 15, 2007, the Request for Reconsideration filed on February 27, 2007, has not been considered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The subject invention is directed towards methods and apparatus for efficiently simulating and testing a chip circuit and an associated code representation. The simulation and testing of the chip circuit is accomplished by breaking down a large instruction set into smaller instruction sets in order to minimize time spent evaluating instruction sets that do not have defects. (See page 7, lines 10-14).

Accordingly, as recited in independent claim 1, a method for simulating a model of a chip circuit includes defining a library of components for a processor (see page 3, lines 20-21). Interconnections for a set of pipelined processors including the processor are defined by analyzing the architectural representation of adjacent processors (see page 3, lines 21-22). A processor circuit is generated by combining the library of components and the interconnections for the set of pipelined processors (see page 3, lines 23-24). A code

representation of a model of the set of pipelined processors is generated (see page 3, lines 24-25). Signals generated by the code representation to signals generated by the processor circuit are compared (see page 3, line 25 through page 4, line 1). If the comparison of the signals is unacceptable an output is generated for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit (see page 4, lines 1-3, Figure 7 and page 14, line 20 to page 15, line 13).

Additionally, as recited in independent claim 6, a method for debugging a processor circuit includes identifying a block level location having an error from a first simulation (see Figure 10, page 18, lines 5-6). A patch is inserted into a thread specific to the block level location of the error (see page 18, lines 10-11). The simulation is executed to determine a signal level location of the error through information generated by the patch (see page 18, lines 11-13). A code representation of a processor associated with the error is corrected. (See page 4, lines 4-9, page 18, lines 15-16).

Still further, as recited in independent claim 9, an apparatus for simulating a model of a chip circuit includes a server in which a simulation program logic is stored. The server is configured to execute the simulation program logic. The simulation program logic includes logic for generating a processor circuit by combining a library of components and defined interconnections for a set of pipelined processors. The interconnections are defined by analyzing the architectural representation of adjacent processors. The simulation program logic further includes logic for generating a code representation of a model of the processor and logic for comparing signals generated by the code representation to signals generated by the processor circuit. If the comparison of the signals is unacceptable, the logic for comparing signals includes logic for generating output for display to identify a cause of the unacceptable comparison of the signals at a block level of the code representation. (see page 4, lines 10-18).

Still further, as recited in independent claim 15, a computer readable medium is provided in which program instructions are stored. (see page 3, lines 8-19). A server of a computer system reads the program instructions in the computer readable medium and performs a method for simulating a model of a chip circuit. The method includes defining a library of components for a processor (see page 3, lines 20-21). Interconnections for a set of pipelined processors including the processor are defined by analyzing the architectural representation of adjacent processors (see page 3, lines 21-22). A processor circuit is generated by combining the library of components and the interconnections for the set of pipelined processors (see page 3, lines 23-24). A code representation of a model of the set of pipelined processors is generated (see page 3, lines 24-25). Signals generated by the code representation to signals generated by the processor circuit are compared (see page 3, line 25 through page 4, line 1). If the comparison of the signals is unacceptable an output is generated for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit (see page 4, lines 1-3, Figure 7 and page 14, line 20 to page 15, line 13).

Accordingly, the methods include defining a library of components for a processor (see page 3, lines 20-21) and defining interconnections for a set of pipelined processors including the processor (see page 3, lines 21-22). A processor circuit is generated by combining the library of components and the interconnections for the set of pipelined processors (see page 3, lines 23-24). A code representation of a model for the set of pipelined processors is generated (see page 3, lines 24-25). A first simulation of the processor circuit is initiated and the signals generated by the code representation are compared to the signals generated by the processor circuit (see page 3, line 25 through page 4, line 1). If the comparison of the signals is acceptable, then the processor circuit is

considered acceptable and is used to design the chip. If the comparison of the signals is unacceptable, the processor circuit is unacceptable and an output for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit is generated (see page 4, lines 1-3, Figure 7 and page 14, line 20 to page 15, line 13).

Once the location of the error or cause of the unacceptable comparison of the signals is identified at the block level (see page 18, lines 5-6), a patch is inserted into a thread specific to the block level location of the error (see page 18, lines 8-10). A second simulation is executed with the inserted patch to determine a signal level location of the error through information generated by the patch that pinpoints the error location (see page 18, lines 11-15). Upon identifying the signal level location of the error, a code representation of a processor associated with the error is corrected (see Figures 5 and 6; related description on page 12, line 3 to page 14, line 8, page 18, lines 15-16).

As referred to in this application, each of the pipelined processors include both hardware such as, input socket interface, star processor, output socket interface, hardware accelerator, and relevant software to access the hardware. (Figure 3 and related description on page 8, line 15 to page 11, line 5). For the pipelined processors the output socket interface of a first processor is in communication with an input socket interface of a second processor, and so on (see Figure 2).

A thread, as used in the application, is a microcode that is executed from start to end. (See page 11, lines 16-18). After power initialization of the processor, the processor is loaded with a basic system microcode thread. This system microcode thread, upon request from a host, can load function specific microcode threads into its code memory. (see page 11, lines 12-15). In order to produce multiple threads the microcode will self-trigger itself, with each microcode executing as one thread. (page 11, lines 16-19). These

specific microcode threads support debugging features that allow the efficient pinpointing of a defect in the processor circuit (page 11, lines 12-16).

The embodiments first execute a simulation at a block level to locate any errors. Therefore, any blocks that are error-free avoid additional testing as is typically performed for a simulation. Any blocks with errors, have a patch inserted to further identify the signal level location of the error within the block so that the error in the design simulation may be corrected.

It should be appreciated that the above description represents only a summary of the present invention. A more in-depth discussion of the present invention is provided in the Detailed Description section of the application.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Shimogori et al. (U.S. Patent Publication No. 2002/0152061 A1) (hereinafter "Shimogori") in view of Shridhar et al. (U.S. Patent No. 5,815,714) (hereinafter "Shridhar").

VII. ARGUMENT

A. Rejection of claims 1-19 under 35 U.S.C. § 103(a) as being unpatentable over Shimogori in view of Shridhar.

Claims 1-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Shimogori in view of Shridhar.

Independent Claims 1, 6, 9 and 15

1. The combination of Shimogori and Shridhar does not teach block level processing and running a second simulation with a patch applied at the block level to further identify a signal level location of the error.

With reference to claim 1, the Applicants disagree with the Examiner's characterizing of the block level processing in Shimogori. The Examiner equates the block level processing of the instant application with parts of C language source code that are converted to VU instructions (conversion to hardware instructions) of Shimogori in order to speed up the execution of the code. By Examiner's own admission, Shimogori does not explicitly suggest or teach block level processing as is evidenced in the Final Office Action page 9, 2nd paragraph "Although Shimogori does not explicitly state, "block level processing....". Identifying the software code that needs to be speeded up does not ***identify a block*** that contains the "error" code (cause of the unacceptable comparison of the signals).

The Examiner is further pointing to the following paragraph to assert that Shimogori teaches "inserting a patch into a ***thread specific to the block level location of the error.***"

> [0090] In this way, the program (C language description) that includes a test function or functions (pseudo-VUs) that operate according to pseudo-VU instructions is verified in step 122 using the ISS system 61. The test functions are ported with the pseudo-VU instructions, so that it is possible to immediately judge whether the test functions have been properly ported in the program that has been compiled by the pcc 101. In the design stage 112, using how many clocks are consumed in what parts of the C language source code reported by the ISS system 61 as an ISS profiler, it is possible to investigate what parts of the code can be speeded up through conversion to VUs (i.e. conversion to hardware). So, which parts of the code are to be realized by VUs is judged.

As can be seen, there is no mention of block level processing occurring in Shimogori. Shimogori teaches identifying a portion of a code that can be speeded up by converting relevant software code to hardware instructions. The identified piece of software code is extracted from the executing function and replaced with a hardware (VU) instruction (patch). By replacing the identified piece of code, the patch takes care of what the Examiner constitutes as an "error."

Further, the Examiner's portrayal of an "error" in Shimogori is not actually an error. The number of clock cycles that a particular piece of code takes, in Shimogori, does not constitute an error that may cause drastic consequences if left undetected. The "error" in Shimogori is a performance delay that is overcome by the "patch." The error, in the claimed invention, is a defect in a chip that when left undetected can cause drastic consequences during the operation of the chip, i.e. the chip will not function. (Page 2, line 8, page 14, lines 3-4, page 18, lines 13-14).

In light of the foregoing discussion, the Shimogori reference does not disclose each and every element of the claimed invention. By Examiner's own admission, Shimogori further does not suggest or teach generating output for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit and has referred to Shridhar to teach inserting a print command to determine the signal level of an error.

Shridhar discloses a process of re-generating debug commands. According to Shridhar, a simulator executes assembled object code in conjunction with a debugger which runs the stored debug commands, as designated, during the execution cycle. Upon termination of the simulation run, the source program is modified (to fix the errors revealed during the debug cycle) and the debug commands are automatically re-generated with correct addresses from the modified source program during assembly of the modified

source code. The re-generation of debug commands ensures that the break-points are relocated to the correct source lines in the modified source program. As in Shimogori, the first simulation run in Shridhar identifies the error which is then corrected by a "patch" that modifies the source code. As can be seen, there is no mention of block level processing occurring in Shridhar. Accordingly, Shridhar does not address the deficiencies of Shimogori with block level processing.

With reference to claim 6 of the instant application, a block level location having an error is identified from a first simulation. A patch is introduced into a thread specific to the block level location of the error and a second simulation is executed. The second simulation allows the patch to provide information that enables determining the signal level location of the error (within the identified block). Once the location of the error is identified, the code representation related to the block of the processor with the identified error is then corrected (Figure 10). Thus, the patch of the claimed invention is used in further identifying the signal level location of the error through information generated by the patch and is not used in correcting a code representation of related processor associated with the error. In other words, the patch of the claimed invention is not to cure the error but to further define the location within the appropriate block level identified by the first simulation. This is in contrast to Shimogori where the "patch" is used to cure what the Examiner refers to as an error by reducing the number of cycles to an acceptable number. The patch of Shimogori is incorporated by replacing parts of the software code with hardware instructions to take care of the excess cycle time taken by a particular portion of the code, which is entirely different from the patch of the claimed invention.

Further, Shimogori does not suggest running the (second) simulation for further identifying the signal level location of the error using the patch. Shimogori's patch inserts hardware instructions that address the excess cycle time problem but will not provide any

relief to the error that caused the unacceptable comparison of signals (in circuitry) of the claimed invention. Shimogori simply does not disclose the claimed invention. By identifying the errors first at the block level and then at the signal level, the claimed invention allows for the errors to be identified faster and corrected more efficiently instead of sifting through a number of error signals and manually determining which block is at issue. (See page 14, lines 4-8, page 16, line 24 - page 17, line 2). Thus, using the methods of the claimed invention, higher level debugging at the block level is achieved which is unrelated to the patch inserting hardware instructions under Shimogori.

The Examiner has failed to meet the Examiner's burden of distinctly identifying each and every element of the claimed invention in the combined references as Shimogori does not disclose having a first simulation to identify an error at a block level of the simulated circuit and applying a patch to further identify a signal level location of the error within the identified block using the information generated by the patch. Rather, Shimogori suggests identifying part of a software code that can be speeded up by converting the software code into hardware instructions. In order to convert the software code to hardware instructions, a part of the software code is identified and extracted and replaced by VU instruction. The modified code including VU instructions, PU instructions and pseudo-VU instructions is compiled by the processor's compiler. The replaced VU instruction would address the excess cycle time problems experienced by the original software code. As the "patch" in Shimogori has already addressed the error in the software code, there is no need for the process of Shimogori to execute the simulation for a second time to further identify the signal level location of the error as is specified in claim 6. The Examiner has provided no explanation to address the execution of the second simulation to determine a signal level location of the error through information generated

by the patch as is evident by the observation made by the Examiner in the Final Office Action on page 9, which reads:

" In regards to claim 6, Applicant is directed to the previous argument made of comparing signals and identifying a cause for unacceptable comparison of signals. As Shimogori identifies a cause of an unacceptable comparison of signals – a cause of too many cycles – by definition, Shimogori identifies an error ([0062]). Also, as Shimogori includes code which will eliminate the cause of unacceptable comparison of signals – which will increase the speed of execution and reduce the number of cycles to an acceptable number – Shimogori inserts a patch into a thread specific to the block level location of the error ([0090])."

Accordingly, Shimogori fails to disclose identifying a block level location having an error from a first simulation, inserting a patch into a thread specific to the block level location of the error, executing the simulation (once more) to determine a signal level location of the error through information generated by the patch and correcting a code representation of a processor associated with the error.

Independent claims 9 and 15 are directed to an apparatus claim that implements the method claim of claim 1. Based on arguments presented with reference to claim 1, the Applicants submit that claims 9 and 15 are patentable over Shimogori.

The Applicants submit that the independent claims 1, 6, 9 and 15 are patentable under 35 U.S.C. § 103(a) over Shimogori in view of Shridhar.

Each of dependent claims 2-5, 7-8, 10-14 and 16-19 ultimately depend from independent claims 1, 6, 9 and 15 respectively. Accordingly, claims 2-5, 7-8, 10-14 and 16-19 are patentable under 35 U.S.C. § 103(a) over Shimogori and Shridhar for at least the same reasons set forth above regarding independent claims 1, 6, 9 and 15.

For at least the foregoing reasons, the Shimogori reference in view of Shridhar does not anticipate the methods and computer readable media of claims 1, 6, 9, and 15. Thus, the rejection of claims 1-19 under 35 U.S.C. § 103(a) as being unpatentable by the Shimogori in view of Shridhar reference is improper and should be reversed.

B. Conclusion

In view of the foregoing reasons, the Applicants submit that each of the claims 1-19 are patentable. Therefore, the Applicants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's rejections of the claims on appeal.

Respectfully submitted,
MARTINE PENILLA & GENCARELLA, LLP

Jayanthi Minisandram

Jayanthi Minisandram
Reg. No. 53,330

710 Lakeway Drive, Suite 200
Sunnyvale, CA 94085
Direct Dial: 408.749.6905
Facsimile: (408) 749-6901
Customer Number 25920

VIII. CLAIMS APPENDIX

1. A method for simulating a model of a chip circuit, comprising method operations of:

defining a library of components for a processor;

defining interconnections for a set of pipelined processors including the processor, the interconnections defined by analyzing the architectural representation of adjacent processors;

generating a processor circuit by combining the library of components and the interconnections for the set of pipelined processors;

generating a code representation of a model of the set of pipelined processors; and

comparing signals generated by the code representation to signals generated by the processor circuit,

wherein if the comparison of the signals is unacceptable, the method includes,

generating output for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit.

2. The method of claim 1, wherein the library of components is included as register transfer logic (RTL).

3. The method of claim 1, wherein the interconnections for the set of pipelined processors is included in a structural netlist.

4. The method of claim 1, wherein the set of pipelined processors are configured to manipulate layers of a header of a data packet in stages.

5. The method of claim 1, wherein the method operation of identifying a cause of the unacceptable comparison of the signals at a block level of the processor circuit includes,

inserting a patch into the code representation to identify a signal level location for the unacceptable comparison of the signals.

6. A method for debugging a processor circuit, comprising:

identifying a block level location having an error from a first simulation;

inserting a patch into a thread specific to the block level location of the error;

executing the simulation to determine a signal level location of the error through information generated by the patch; and

correcting a code representation of a processor associated with the error.

7. The method of claim 6, wherein the patch is a print command.

8. The method of claim 6, wherein the method operation of executing the simulation to determine a signal level location through information generated by the patch includes,

triggering a print statement indicating the signal level location of the error.

9. An apparatus for simulating a model of a chip circuit, comprising:

a server in which a simulation program logic is stored, the server configured to execute the simulation program logic, wherein the simulation program logic includes:

logic for generating a processor circuit by combining a library of components and defined interconnections for a set of pipelined processors, the interconnections defined by analyzing the architectural representation of adjacent processors;

logic for generating a code representation of a model of the processor; and

logic for comparing signals generated by the code representation to signals generated by the processor circuit,

wherein if the comparison of the signals is unacceptable, the logic for comparing signals includes,

logic for generating output for display to identify a cause of the unacceptable comparison of the signals at a block level of the code representation.

10. The apparatus of claim 9, wherein logic for identifying a cause of the unacceptable comparison of the signals at a block level of the processor circuit includes,

logic for inserting a patch into the code representation to identify a signal level location for the unacceptable comparison of the signals.

11. The apparatus of claim 9, wherein the library of components is included as register transfer logic (RTL).

12. The apparatus of claim 9, wherein the interconnections for the set of pipelined processors is included in a structural netlist.

13. The apparatus of claim 10, wherein the patch includes logic for executing a print statement.

14. The apparatus of claim 9, wherein each logic component is one of hardware and software.

15. A computer readable medium in which program instructions are stored, the program instructions when read by a server of a computing system, cause the server to perform a method for simulating a model of a chip circuit, the method comprising:

defining a library of components for a processor;

defining interconnections for a set of pipelined processors including the processor, the interconnections defined by analyzing the architectural representation of adjacent processors;

generating a processor circuit by combining the library of components and the interconnections for the set of pipelined processors;

generating a code representation of a model of the set of pipelined processors;

comparing signals generated by the code representation to signals generated by the processor circuit,

wherein if the comparison of the signals is unacceptable, the method includes,

generating output for display to identify a cause of the unacceptable comparison of the signals at a block level of the processor circuit.

16. The computer readable medium of claim 15, wherein the library of components is included as register transfer logic (RTL).

17. The computer readable medium of claim 15, wherein the interconnections for the set of pipelined processors are included in a structural netlist.

18. The computer readable medium of claim 15, wherein the set of pipelined processors are configured to manipulate layers of a header of a data packet in stages.

19. The computer readable medium of claim 15, wherein identifying a cause of the unacceptable comparison of the signals at a block level of the processor circuit includes,

inserting a patch into the code representation to identify a signal level location for the unacceptable comparison of the signals.

IX. EVIDENCE APPENDIX

There is currently no evidence entered and relied upon in this Appeal.

X. RELATED PROCEEDINGS APPENDIX

There are currently no decisions rendered by a court or the Board in any proceeding identified in the Related Appeals and Interferences section.